Aix–Marseille Université
Faculté des Sciences
Master M1 MAS
Prof. Kai Schneider

1st semester 2023/2024

TP-2 Representation and approximation of structured data using R

Graphics in R

There are many possibilities for generating graphics in R . One distinguishes between graphics functions at two levels:

- high level functions of the type ’plot’, ’boxplot’, ’hist’, ... which generate a new figure.
- low level functions, which allow to add objects to a graphics and to refine it.

As example we will test the following lines of code given below, which plot in the plane two vectors (x_1, y_1) and (x_2, y_2), generated by pseudo-random numbers with given mean and standard deviation, using colors and symbols. The corresponding graphics is shown in figure 1.

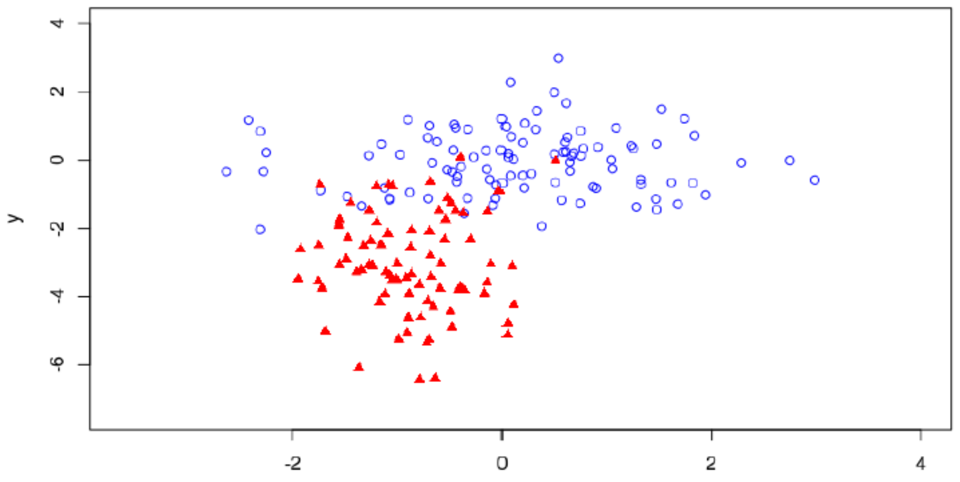

Figure 1: An example of a scatter plot.

```
> x1 = rnorm(100)
> y1 = rnorm(100)
> x2 = rnorm(80,mean=-1,sd=0.5)
> y2 = rnorm(80,mean=-3,sd=1.5)
> xrange = range(c(x1,x2))
> yrange = range(c(y1,y2))
> plot(x1,y1,xlim = xrange + c(-1,1),ylim = yrange + c(-1,1),col=’blue’,xlab=’x’,ylab=’y’)
> points(x2,y2,col=’red’,pch=17)
```

The vectors ’xrange’ and ’yrange’ define the range of values of x and y to be shown (they enter ’xlim’ and ’ylim’ and are automatically determined), ’col’ selects the color of the points and ’pch’ the symbols, ... A large number of parameters can be adjusted, we refer to the online documentation.

Exercise 1: graphics

1. Plot the function 'sin' on the interval $[0, 10]$ imposing the limits $[-.15, 1.5]$ for the y-axis (option 'ylim') and put as title 'Graphics of the sinus function', the name 'Angle' on the x-axis and the name 'Sinus' on the y-axis.

2. Add the line given by $y = x/(2\pi)$ onto the graphics using the function 'lines', in red color. Using 'points', add the coordinates of the points (0, 1.3) and (2,-1) using the symbol '$\star$' in green.

As mentioned R offers many possibilities for plotting graphics. For example it is possible to separate a graphics window into sub-windows using the instruction 'par(mfrow=c(nb-lines, nb-columns))', as shown in the example below, which illustrates a histogram and a boxplot of a random vector in the same window, cf. figure 2.

```
> u = rnorm(1000,mean=3,sd=5)
> par(mfrow=c(1,2))
> hist(u)
> boxplot(u)
> help(boxplot)
> title('Boxplot')
```

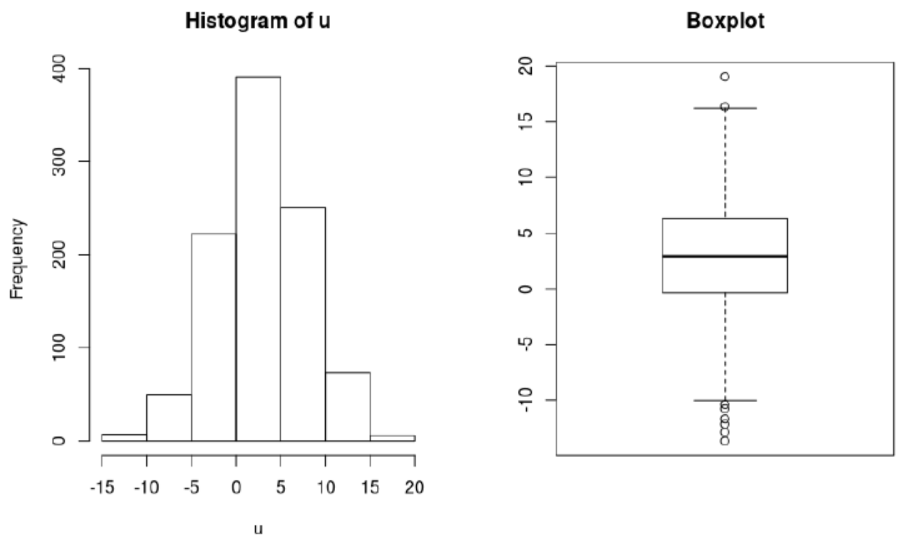

Figure 2: Splitting a graphics window into two sub-windows.

Programming, scripts and functions under R

Control structures
Conditional execution using 'if', 'else' and 'switch' allows that a command is executed if a certain condition is satified. The syntax is the following:
'if (condition) code.true else code.false'
where 'code.true' and 'code.false' represent each an instruction or a series of instructions, in the latter those need to be written in parentheses.

```
> x = -5
> if (x>0) print('x positif') else {x = -x; print('x negatif, je change son signe')}
[1] "x negatif, je change son signe"
```

Loops
In many cases it is advantageous to avoid loops, because they are typically slower than operations which can be expressed in vector form. The most common loop is the 'for' loop, and the syntax is:
    'for (variable in series) series of instructions'
In case there is more than one instruction they have to been placed in parentheses. Here 'series' is a series of values which can be given by 'variable' (and hence they are not necessarily ordered numbers, even not numbers). The following lines below give a simple computation of the factorial of 5.

```
> k = 5
> fact_k = 1
> for (n in 2:k) fact_k = fact_k * n
> fact_k
[1] 120
```

An alternative is the loop using 'while', where the syntax is given by
    'while (condition) series of instructions'

```
> k = 5
> n = 1
> fact_k = 1
> while (n <= 5) {fact_k = fact_k*n; n = n + 1}
> fact_k
[1] 120
```

Please also check 'switch'. Other possibilities are the instructions 'repeat' and 'break', which are less used.

Exercise 2: loops

1. Using the 'for' loop compute the sum of the first 100 integer numbers and the sum of their squares.

2. Do the same using the while loop.

Use of scripts

In the console instructions are entered line by line and it is thus not possible to have a global vision of a working session. Scripts allow entering directly a sequence of instructions, saving them (in a text file) to be able to execute them, modify them or run them again.

The development environment of RStudio (like the majority of other R environments), allows to manipulate, save and execute scripts. Executing a script is done via the function 'source'.

Use of functions
In general it can be useful to assemble a series of instructions frequently used under the form of a function. A function is defined by the following syntax:
    'my-function= function(arguments) expression'
where 'my-function' is the name of the function, 'arguments' is a list of arguments, separated by commas and 'expression' is the content of the function, either a simple expression, or a group of expressions assembled within parentheses. The following example is a function which computes the factorial of an integer number.

```
ma_factorielle = function(k){
  if (k-floor(k)!=0) stop('argument non entier')
  if (k<0) stop('argument negatif')
  if (k == 0 | k==1)
    fact_k = 1
  else{
    fact_k = 1
    for (n in 2:k)
      fact_k = fact_k * n
    }
  fact_k
}
```

The first lines check if the argument is a positiv integer or equal to zero.

A function is saved into a text file (which can be generated in RStudio with the integrated text editor), for example my-factorial.R. The function 'source('my-factorial.R')' allows to get an executable function. This becomes necessary after each modification of the file. Note that a text file can have several functions.

A function yields simply the result of the last expression in the body of the function, in the example below the result of computing the factorial:

```
> ma_factorielle(3)
[1] 6
> ma_factorielle(-3)
Error in ma_factorielle(-3) : argument negatif
```

In the case where the function returns several results, the simplest solution is to organize them into a list. For example the function below returns the mean and the median of a vector.

```
mm = function(u){
  a = mean(u)
  b = median(u)

  list(mo=a,me=b)
}
```

The execution of the above function yields the following result:

```
> u=rnorm(100,mean=1)
> v = mm(u)
> v
$mo
[1] 0.8662206
$me
[1] 0.7833275
> v$mo
[1] 0.8662206
```

Exercise 3: functions in R

Write a function called 'mention' which has as argument a mark between 0 and 20 and which returns the corresponding mention: A (0-10^-), P (10-12^-), AB (12-14^-), B (14-16^-), TB (16-18^-), TTB (18-20). An error message shall be given if the mark is not correct, i.e. for negative values or values larger than 20.

Installing and downloading packages in R

A package in R is a library of functions for specific subjects. To be able to use a package it needs to be installed and then loaded. A large number of packages is available for R. Some are already included in the basic distribution, other are intalled and loaded by default in RStudio.

Installing a package is done with the function 'install-packages' in the command line or using the tab 'packages' in RStudio. Loading a package at the command line is done with the function 'library'. For example try to install and to load the package 'gplot' which yields advanced graphics tools.

Further information and manuals are avaiable for R, in particular we refer to
https://cran.r-project.org/manual.html

Discrete Fourier transform (DFT) and FFT in R

We consider the linear operator $F : \mathbb{C}^N \to \mathbb{C}^N$ defined by

$$X_k = (Fx)_k = \sum_{n=0}^{N-1} x_n \, W_N^{kn}$$

for $k = 0, 1, ..., N-1$ and $x \in \mathbb{C}^N$, where $W_N^{kn} = e^{-i(2\pi/N)kn}$ and $i = \sqrt{-1}$. The vector $X_k = (Fx)_k \in C^N$ is called the discrete Fourier transform (DFT) of x.

The inverse discrete Fourier transform (IDFT) is defined by

$$x_n = (F^{-1}X)_n = \frac{1}{N} \sum_{k=0}^{N-1} X_k \, W_N^{-kn}$$

for $n = 0, 1, ..., N-1$ and $X \in \mathbb{C}^N$

For computing the DFT (and its inverse) in R we use the command 'fft' (and 'fft(X.k, inverse=TRUE) / length(X.k)'). To get a detailed description use 'help(fft)'.

Exercise 4:

We consider the signal $s(t) = \exp(-(t-t_0)^2/\sigma^2)$ and the discrete signal $s_n = s(t_n)$ with $t_n = n/N$ for $n = 0, ..., N-1$ and with $t_0 = 0.5$, $\sigma^2 = 1/500$ and $N = 2^{10} = 1024$.

- Plot the discrete signal s_n.
- Compute its discrete Fourier transform

$$\widehat{s}_k = \frac{1}{N} \sum_{n=0}^{N-1} s_n \exp(-i2\pi kn/N) \quad , \quad k = 0, ..., N-1$$

  (e.g. using fft(s))

- Plot the spectrum (modulus of the Fourier coefficients using e.g. 'plot.frequency.spectrum()'), $|\widehat{s}_k|, \; k = 0, ..., N-1$.

Exercise 5:

Given a discrete Dirac pulse

$$d_n = \begin{cases} 1 & \text{for } n = mN/f_0 \,, \quad m \in I\!N \\ 0 & \text{else} \end{cases}$$

for $n = 0, ..., N-1$ with $N = 2^{10} = 1024$, and frequency $f_0 = 2^6 = 64$.

- Plot the Dirac pulse.
- Compute its discrete Fourier transform.
- Plot the spectrum.

Exercise 6:

Multiply the signal s with the Dirac pulse and plot the resulting sampled signal $r_n = s_n d_n$ for $n = 0, ..., N-1$.

- Plot the spectrum of r.
- What do you observe?

Exercise 7:

Reconstruct the signal s^{rec} from the sampled signal r by applying an ideal low pass filter in Fourier space with cut off frequency $k_c = f_0/2 = 32$, i.e.

$$\widehat{s}_k^{rec} = \begin{cases} \widehat{r}_k & \text{for } |k| \leq k_c \\ 0 & \text{else} \end{cases}$$

for $k = 0, ..., N-1$.

Apply the inverse discrete Fourier transform (e.g. using fft(s^{rec}, inverse=TRUE) / length(s^{rec})) to get

$$s_n^{rec} = \sum_{k=0}^{N-1} \widehat{s}_k^{rec} \exp(i2\pi kn/N) \quad , \quad n = 0, ..., N-1$$

and plot s^{rec}.

Exercise 8:

Repeat exercises 5-7 with different values $f_0 > 64$ and $f_0 < 64$.
What do you observe?